University of Pittsburgh

Algorithms and Data Structures 2
CS 1501

Spring 2023

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming Deadlines
  - Homework 5: this Friday @ 11:59 pm
  - Lab 3: Tuesday 2/14 @ 11:59 pm
  - Assignment 1: Friday 2/17 @ 11:59 pm

Previous lecture

- Digital Searching Problem
  - What if we use the fact that data items are represented as bits in computer memory?
- Digital Search Tree (DST)

This Lecture

- Digital Search Tree (DST)
- Radix Search Trie (RST)

Digital Search Trees (DSTs)

Instead of looking at less than/greater than, lets go left or right based on the bits of the key

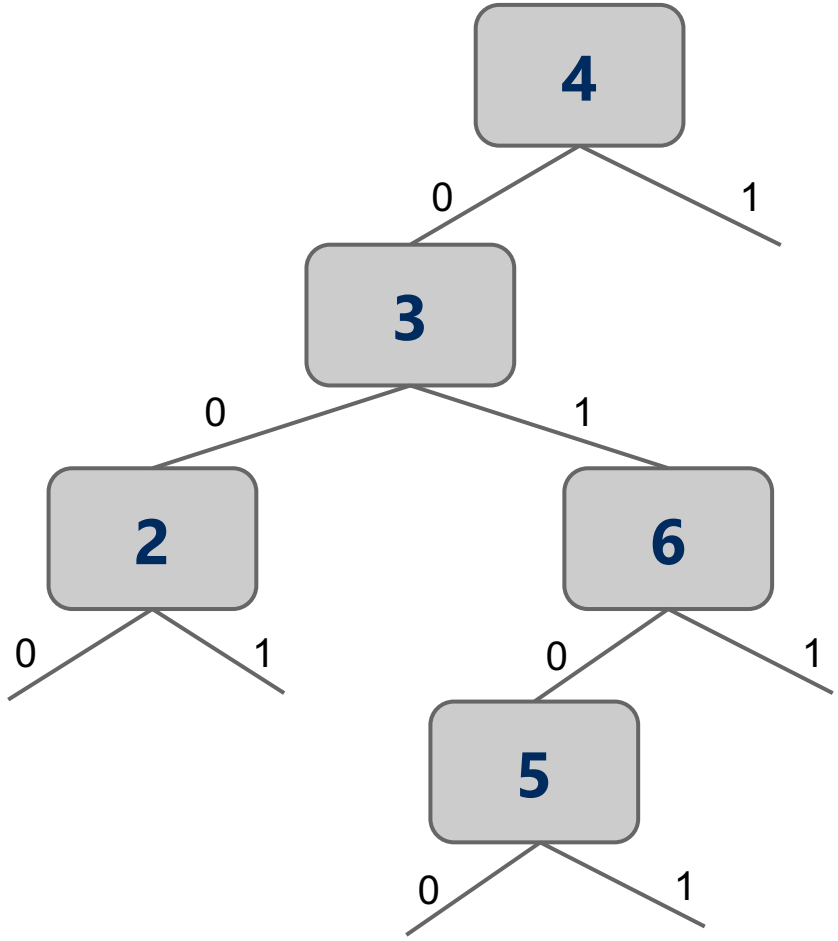
DST example: Insert and Search

Insert:

4	0100
3	0011
2	0010
6	0110
5	0101

Search:

3	0011
7	0111



Inserting into a DST

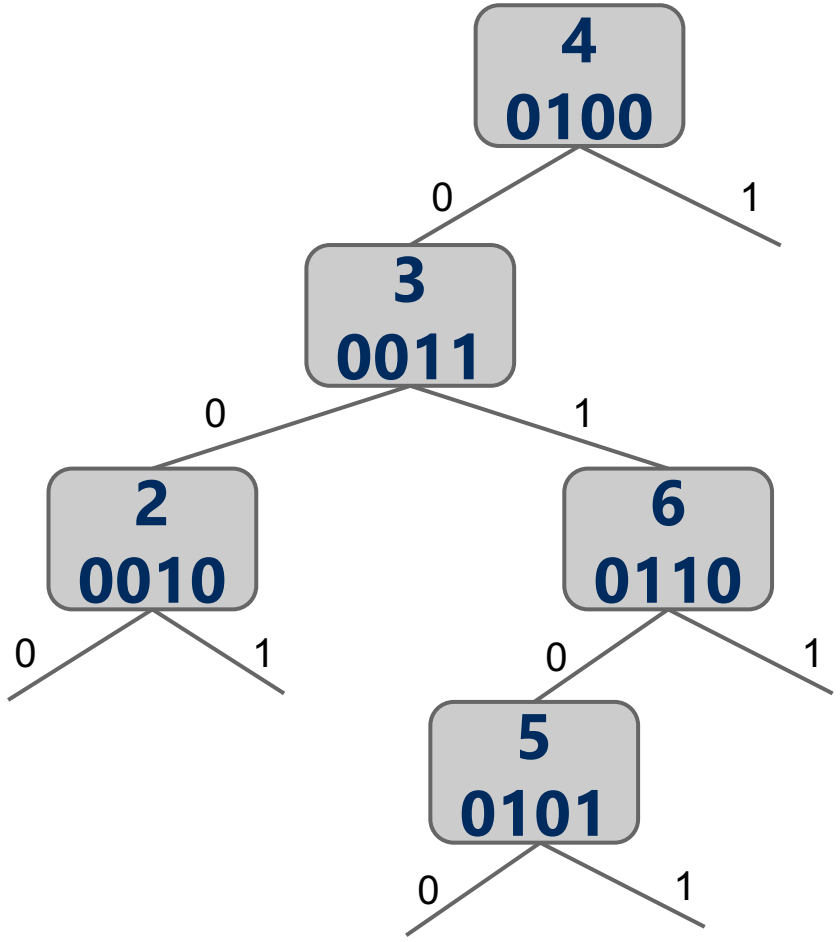
- adding a key *k* and a corresponding value
  - if root is null, add *k* as the root and return
  - current ← root
  - Repeat
    - if k is **equal to** the current.key, replace value and return
    - if current bit of k is 0,
      - if left child is null, add *k* as left child
      - else continue to left child (recursive call or current ← current.left)
    - if current bit of k is 1,
      - if left child is null, add *k* as right child
      - else continue to right child (recursive or current ← current.right)
- When does the algorithm stop?
  - no more bits or
  - hitting a null

Runtime Analysis of Digital Search Trees

- b: the bit length of the target or inserted key
- n: number of nodes in the tree
- Worst-case Runtime?
  - min(b, height of the tree)
- What is the average height of the tree?
  - Assume that having 0 or 1 is equally likely at each bit
  - log(n)
  - In general b >= $\lceil \log n \rceil$

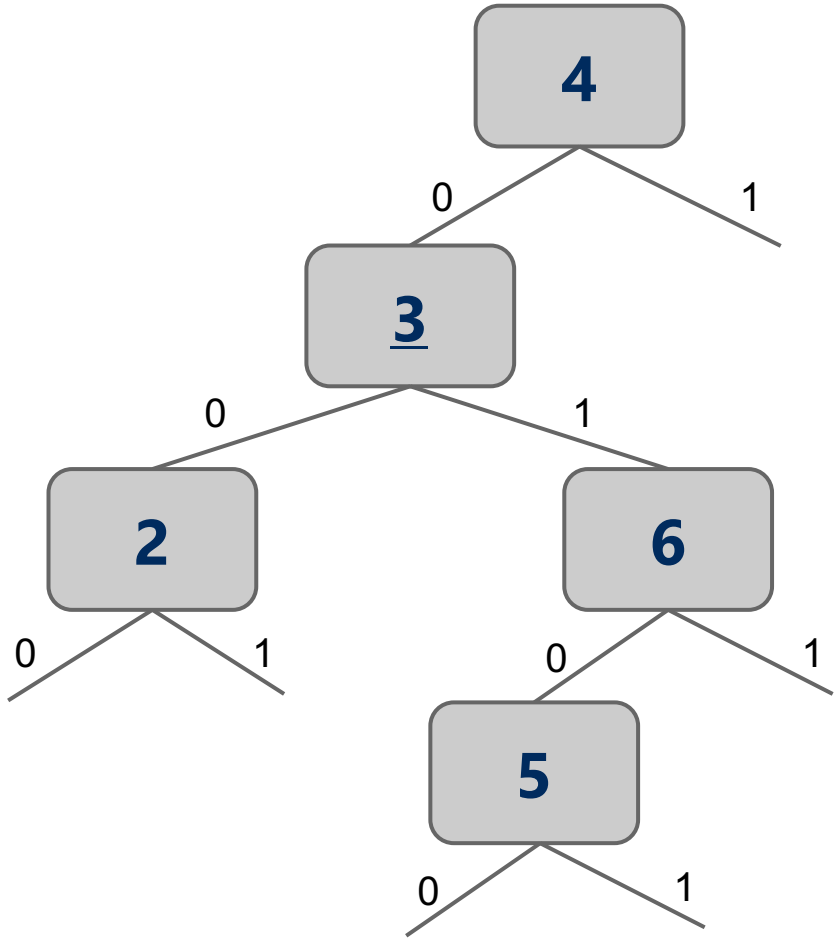
What property does a DST hold?

- In a DST, each node shares a **common prefix of length depth(node)** with all nodes in its subtree
  - e.g., 6 shares the prefix "01" with 5
- In-order traversal doesn't produce a sorted order of the items
  - Insertion algorithm can be modified to make a DST a BST at the same time



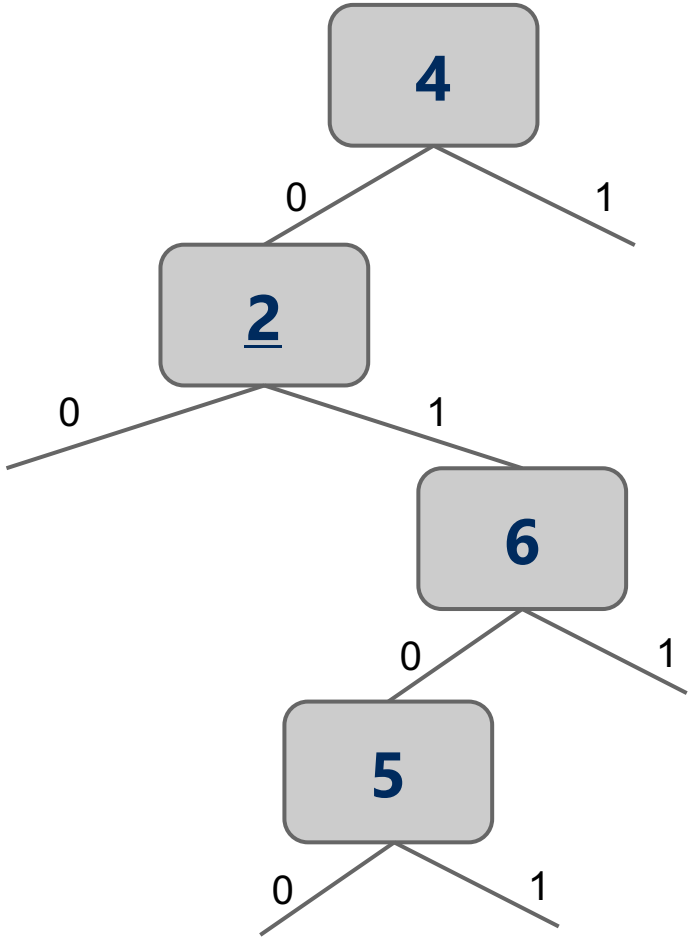
DST example: Delete

- Delete 3
- Can replace it with any leaf in its subtree
- Let's replace it with 2
- OK because 2 shares "0" as a prefix with 3, so it also shares "0" as a prefix with 6 and 5



DST example: Delete

- Delete 3
- Can replace it with any leaf in its subtree
- Let's replace it with 2
- OK because 2 shares "0" as a prefix with 3, so it also shares "0" as a prefix with 6 and 5

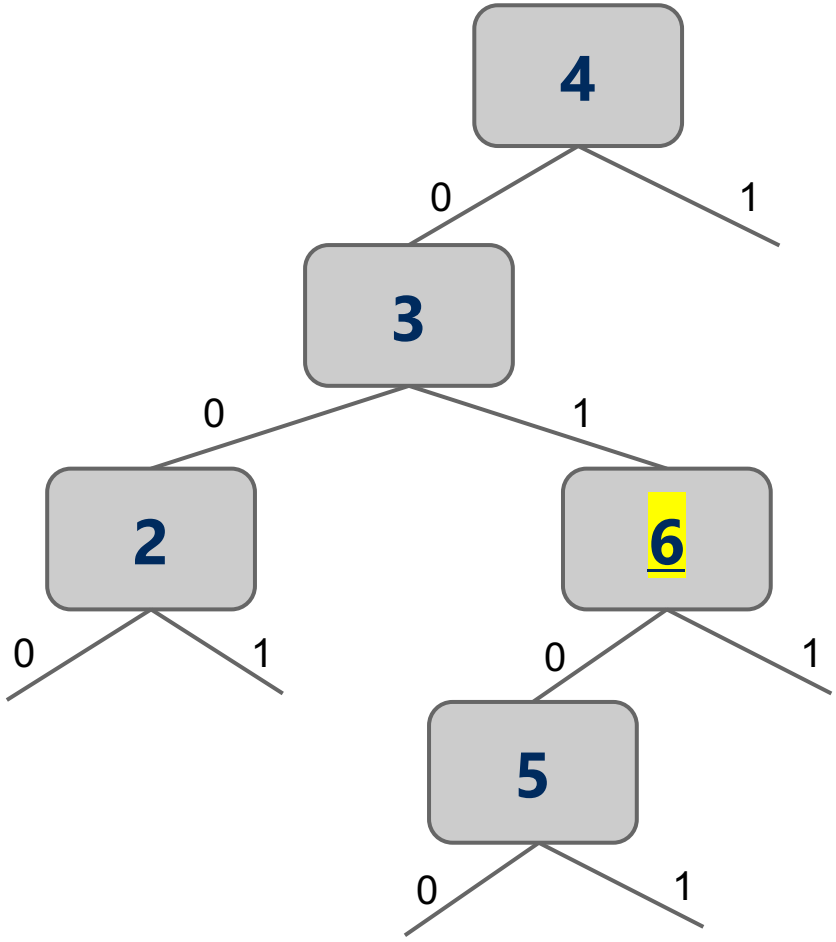


DST example: Variable length keys

- Insert

1 01

- Must be in place of 6
- Replace 6 by 1 and re-insert 6



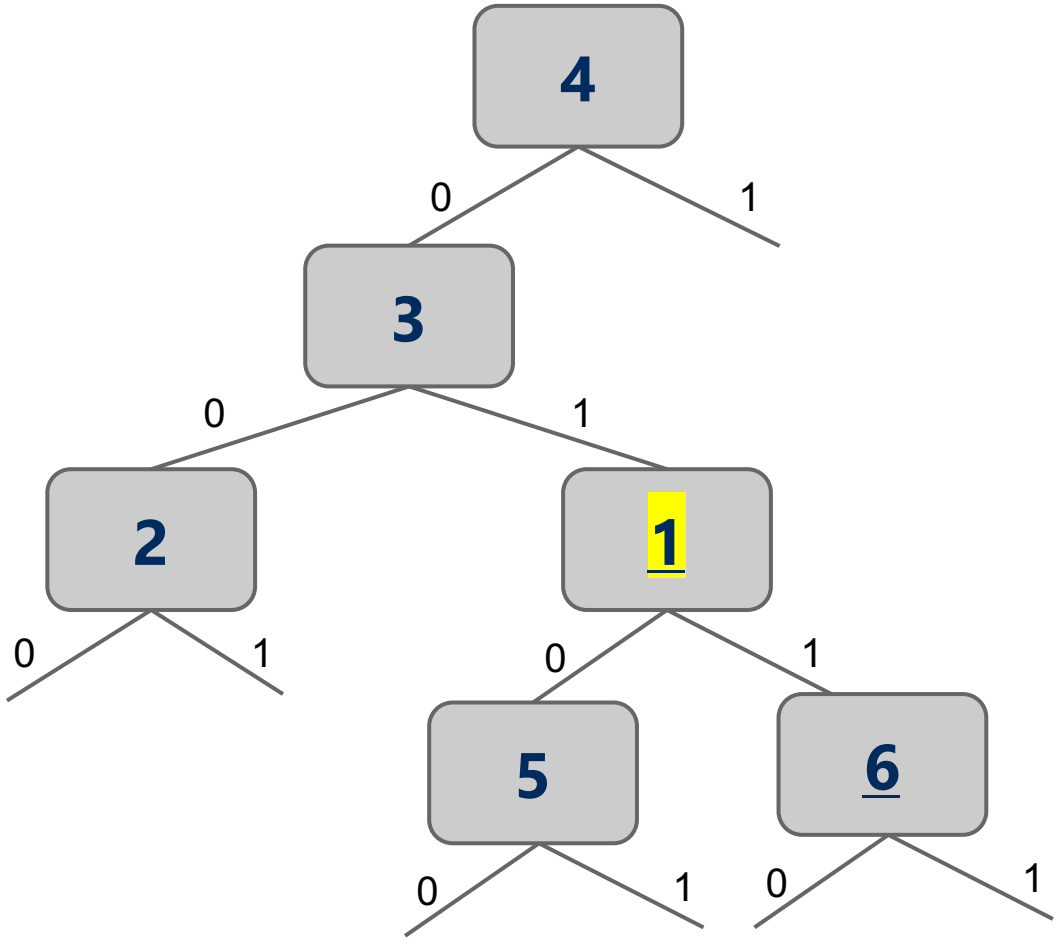
DST example: Variable length keys

- Insert

1 01

- Must be in place of 6
- Replace 6 by 1 and re-insert

6 0110



Analysis of Digital Search Trees

- We end up doing many **equality** comparisons against the full key
- This is better than less than/greater than comparison in BST
- Can we improve on this?

Radix search tries (RSTs)

- Trie as in retrieve, pronounced the same as "try"
- Instead of storing keys inside nodes in the tree, we store them implicitly as paths down the tree
  - Interior nodes of the tree only serve to direct us according to the bitstring of the key
  - Values can then be stored at the end of key's bitstring path (i.e., at leaves)
  - RST uses less space than BST and DST

Adding to Radix Search Trie (RST)

- Input: key and corresponding value
- if root is null, set root ← new node
- current node ← root
- for each *bit* in the key
  - if bit == 0,
    - if current.left is null, set current.left = new node
    - move to left child
      - set current ← current.left
  - if bit == 1,
    - if current.right is null, set current.right = new node

    current ← current.right
- current.value = value

RST example

Insert:

4 0100

3 0011

2 0010

6 0110

5 0101